51. (new) The method of claim 50, wherein the control of control transfer behavior includes transfer of execution control to a second instruction for execution, the second instruction being coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

52. (new) The method of claim 50,

wherein entries of the table correspond to pages managed by a virtual memory manager, circuitry for locating a table entry being integrated with virtual memory address translation circuitry of the computer.

54. (new) The method of claim 50, further comprising the steps of:

triggering an interrupt on execution of an instruction of a process, synchronously based at least in part on a memory state of the computer and the address of the instruction, when the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

55. (new) The method of claim 50, wherein:

the table entries are indexed by a virtual address of the instruction.

56. (new) The method of claim 50, wherein:

the table entries are indexed by a physical address of the instruction.

57. (new) The method of claim 24, further comprising the step of:

after determining that the existence of an alternate coding is likely, consulting a second table, the entries of the second table definitively indicating entry points for initiating such alternate codings as exist.

REMARKS

Claims 1-57 are pending in the present application, a total of 57 claims. Claims 1, 2, 10, 14, 19, 24, 30, 39 and 50 are the independent claims under consideration.

The amendments to the existing claims are made to clarify issues for the Examiner, generally without narrowing the claims. Applicant has also added new claims 36-57 to cover additional aspects of the invention that are fully supported in the specification. No new matter has been added.

Applicant respectfully requests reconsideration of the application.

I. Introduction

Among other purposes, the inventions recited in the claims are useful to enable use of two different instruction set architectures (ISA's) in a single computer, in a way that allows emulation of an "old" instruction set (e.g., the Intel X86 instruction set) in a computer with a "new" instruction set (e.g., a RISC instruction set). One of the requirements for reliable emulation is that the program text may not be altered by the emulation process. For example, if the X86 program uses self-modifying code, any alteration of the X86 instruction text by the emulator may cause the emulation to deviate from the behavior observed if the program were executed on a native X86 computer.

The overall system described in the application (the TAXi system) includes a "hot spot" detector that detects regions of X86 code that are frequently executed. The hot spots are then translated by a binary translator into RISC code. Many of the claims relate to techniques for (i) recognizing when X86 execution has reached a hot spot for which a translation exists, when the instruction text itself may not be changed, (ii) changing the ISA to execute RISC code, and then (iii) transferring control back to the X86 code at the point corresponding to the end of the translated hot spot, all without altering the original X86 program text.

The specification discusses several different "side tables" that are used to implement these inventions. For example, one side table (ISA bits 180, 182) has entries that each correspond to a region of memory, telling whether instructions in the region are to be interpreted in the X86 ISA or in the RISC ISA. A second side table (PFAT 172, 174 with its probe bits 624) has entries that each correspond to regions of X86 instructions. The probe bits 624 of a single PFAT entry give an approximate indication, a likelihood estimate, of whether there is a translated code segment for any of the code in the region corresponding to the PFAT entry.[1] A

[1] In some embodiments, the ISA bits 180 and probe bits 624 may be stored together in a single table. In others, they might be stored separately.

third side table (PIPM 602) is consulted when the PFAT table access indicates that existence of translated code is likely. The PIPM then gives the final definitive answer of whether execution should be transferred to RISC code, and where the relevant RISC code is located in memory.

The TAXi system uses interrupts (i) to seize control of the execution of a program, (ii) to transfer to control from the X86 code to the translated RISC code, and then (iii) to transfer control back to the appropriate point in the X86 code at the end of the translated segment. Some interrupts perform conventional functions like page faults or handling other execution exceptions, while others perform unconventional functions like changing the ISA in which the next instructions will be executed. Some interrupts are generated by instructions like TRAP, that are architecturally-defined to generate interrupts, and some interrupts are generated on simple instructions like ADD's or JUMP's that are not architecturally-defined to generate an exception. For example, if a translated hot spot starts with a simple integer ADD instruction (the definition in the X86 architecture of an integer ADD instruction does not call for an interrupt to be raised), the side tables are set to values that cause this otherwise-interrupt-free instruction to raise an interrupt, so that the TAXi/Tapestry system can gain control and transfer execution over to the translated RISC code.

The Examiner should note that several of the claims of this application are directed to specific computer design techniques that have broad applicability, beyond the specific context described above. Thus, unless a claim specifically recites "two distinct ISA's," or emulation of a non-native ISA, it should be understood that the claim is not so limited. The discussion in this preliminary section is merely to provide context for understanding the technology, not to limit the scope of the claims.

II. The specification

The Examiner indicated that an amendment previously requested at page 62, lines 14-15, replacing "the operating system" with --Tapestry operating system 312-- has not been entered because the text is not at the location. By this paper, the error is corrected by requesting the amendment at page 62, lines 16-17. Accordingly, Applicant believes that this issue is now resolved.

The Examiner indicated that he is unable to find what "TAXi" stands for in the specification. The Examiner's attention is drawn to page 36, lines 8-9 of the specification, where "TAXi" is identified simply as the name for one commercial implementation.

D

III. Wording issues

The Office Action raises a number of questions relating to wording issues. Each of the specific issues is discussed in detail below. As will be shown, Applicant believes that only the issues discussed in sections III.N and III.Q of these Remarks raised proper rejections under § 112. Thus Applicant submits that all amendments (except the amendments discussed in these two sections) are made merely to assist the Examiner's understanding, not pursuant to any statutory requirement. Applicant believes that the application as now amended fully complies with § 112, and requests withdrawal of any rejections that may be outstanding.

A. "table which stores the entries"

The Action questions whether claim 1 might be "incomplete for omitting essential elements," based on an alleged "gap between the elements." According to the Examiner, the omitted element is a table which stores the entries assessed and used by the table lookup circuitry and by the handler respectively.

First, Applicant believes that the claims are drafted in full conformance with MPEP § 2172.01. The specification at no point states that the "table" is "essential to the invention." Applicant relied on § 2172.01 in choosing not to recite the table as a positive element of the claim. Without some showing in the Action that the "table" is "disclosed to be essential to the invention," MPEP § 2172.01 is not properly applied to the claims.

Second, Applicant observes that the "table" is clearly (though inferentially) recited in line 3 of claim 1, as follows:

> table lookup circuitry designed to retrieve an entry from a table, each entry of the table being associated with a corresponding address range translated by address translation circuitry of the microprocessor chip, each entry describing a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range...

Finally, Applicant observes that this is an entirely appropriate use of inferential claiming. The specification describes several embodiments of tables that are off-chip (e.g., ISA bits 180, XP bit 184 or PFAT 172), and several that are on-chip (the copy of ISA bits 182, of XP bits 186 in the I-TLB, or the page properties 624). Further, these tables generally do not exist at the time that the "microprocessor chip" (see preamble of claim 1) is sold. Thus, it would be inappropriate to recite the "table" as a positive element of claim 1.

B. "what event causes the interrupt"

The Action states that "Claim 1 fails to recite what event causes the interrupt." As amended, claims 1, 9, 12, 18 and 29 recite "interrupt criteria" that trigger the interrupt. In different claims, these "interrupt criteria" include, *inter alia*, "the memory state of the computer," "the table entry associated with the address of the instruction," and "the address of the instruction," each of which are fully supported in the specification.[2]

One particular example of how the interrupt is caused is shown in the logic of Fig. 6b and the process shown in Fig. 6c, and discussed in sections VI.D and VI.E at pages 106-112 of the specification. This amendment is believed to clarify the issues raised by the examiner without narrowing the scope of the claims. No new matter has been added.

C. "handler is unable to make a decision"

The Office Action states that "with respect to claim 1, it appears that the handler is unable to make a decision as to whether to effect control of an architecturally-visible data manipulation behavior or if a control transfer behavior of an instruction based on an entry describing a likelihood of the existence of an alternate coding of the instruction. There is no apparent relationship between likelihood of existence and different behaviors."

As amended, claim 1 recites:

> ... the handler being responsive to a content of the table entry to affect the instruction pipeline circuitry to effect control of an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on the contents of a table entry associated with the address range in which the instruction lies

Accordingly, claim 1 now clearly recites that the handler consults the contents of the table, and based on the table contents, decides whether or not to "effect control of ... behavior" and/or decides what behavior to affect. The affect on behavior is achieved by controlling the pipeline circuitry. This clarification is believed to overcome any objection raised by the Examiner.

---

[2] Because the "interrupt criteria" exist only to give a name to the previous limitations of the claim, adding recitation of the "interrupt criteria" is not a narrowing amendment. Similarly, the reorganization of various limitations in the claims are not narrowing amendments.

D

D. "steps for translating addresses, etc."

The Examiner requested information as to "further method steps in regard to the memory structure recited therein" as recited in claim 8.

Applicant respectfully submits that the claim as drafted is entirely proper. Claim 8 recites a limitation on the apparatus that performs a method. Such limitations are entirely proper and conventional. See, for example, the preamble of claim 1, and claims 2 and 8 of the Woods '032 patent, and the latter portion of the "wherein said computer includes" portion of claim 4 of the Morley '982 patent. Applicant intends that the performance of the steps of claim 2, using an apparatus as specified in claim 8, be within the claim, whether or not the apparatus "accesses pages" or "accesses entries within the page." Applicant intends that any method that avoids either the steps of claim 2 or the additional limitations of claim 8 be outside claim 8. Accordingly, Applicant believes that claims 8 and 27 fully comply with § 112, even without amendment.

E. "architectural definition ... not calling for an interrupt"

The Examiner requested information as to the meaning of "the architectural definition of the instruction not calling for an interrupt" as used in claim 1.

An instruction "wherein the architectural definition of the instruction ... does not call for an interrupt" is an instruction that the architecture defines as executing without raising an interrupt. For example, in most computers, the definition of an integer add of two integer registers, where one register contains "2" and the other contains "3," does not call for an interrupt. On the other hand, as is well-known in the art, the definition of an SVC or TRAP instruction calls for an interrupt. In most architectures, a divide instruction, in cases where the divisor is zero, calls for an interrupt. Most architectures define that an access to an undefined memory location calls for an interrupt. Applicant believes that this concept is well known to those skilled in the art.

Some examples of an instruction "wherein the architectural definition of the instruction ... does not call for an interrupt" are discussed in section VI.A (page 100-101 of the specification), section VI.B (pages 101-103), many of the probeable events 610 in Fig. 4b, and many of the X86 transfer of control instructions discussed in section VI.D (pages 106-111).

D

F. "architecturally-visible behavior"

The Examiner requested information as to the meaning of "an architecturally-visible data manipulation behavior or control transfer behavior of an instruction" as used in claim 1.

"Architecturally-visible data manipulation behaviors or control transfer behaviors of an instruction" are discussed in the documents incorporated by reference into the specification at pages 138-139. As is well-known in the art, "architecturally visible behaviors" are behaviors that must be preserved across all implementations of an architecture. For example, the bit sequence "0000 0100" means "add an 8-bit immediate to the A register" in all implementations of the x86 architecture, from the 8086 in the mid-1970's to the Pentium III. On the other hand, pipeline hazard controls, the internal sequencing of a repeated string instruction, and the management of hardware resources during intermediate pipeline stages are not architecturally visible in most architectures. Most architecturally-visible results persist across instruction boundaries, most non-architecturally-visible results do not.

Some instructions have "architecturally-visible data manipulation behavior" – for example, an ADD instruction calls for an addition of data, and all implementations of the architecture must achieve the same result. Other simple data manipulations include subtraction, multiplication, division, negation, logical and, logical or, logical exclusive or, complement, shift, bit field insert or extract, most format conversions, and most floating-point arithmetic operations, etc. ("Data manipulation" behavior can be contrasted to "data movement" behavior of a memory load or store)

An instruction may have a "architecturally-visible control transfer behavior" – for example, a JUMP instruction calls for a control transfer to a particular program location, and all implementations of the architecture must achieve the same result.

Applicant believes that these concepts are well known in the art and that no amendments are necessary to address the issues raised by the Examiner.

Some examples of controlling architecturally-visible instruction behavior based on table entries are discussed in section VI.A (page 100-101 of the specification), section VI.B (pages 101-103), and section VI.D (pages 106-111). The specification may include other examples.

G. "non-supervisor mode program"

The Examiner requested information as to the meaning of "non-supervisor mode program" claim 1.

D

"Non-supervisor mode" is believed by Applicant to be a well-known term of art, not requiring definition in the specification. Roughly-equivalent terms of art include "non-supervisor mode," "problem state," "user mode," "non-privileged mode" and "non-ring zero." The term "non-supervisor mode" is used in the specification at page 18, line 18. The specification has many examples of non-supervisor mode programs, including threads 302 and 304 of Fig. 3a.

H. "different instruction"

The Examiner requested information as to the meaning of "different instruction" in claim 3.

By this paper, claim 3 is amended to recite a "second instruction" instead of a different instruction. Applicant believes that the "second instruction" of claim 3 as amended, now clearly stands in contrast to the "instruction" of claim 2.

Applicant submits that the change from "different" to "second" is not a narrowing amendments.

I. "the content of a table entry"

The Examiner requested information as to the meaning of "the content of a table entry" in claim 2.

By this paper, Applicant has amended claim 2 to recite "a content of a table entry."[3]

J. "control of behavior per se"

The Examiner requested information as to the relationship between "changing control of behavior per se" and "changing ISA" in claim 5.

Claim 5 recites as follows:

> 5. The method of claim 2, wherein the control of architecturally-visible data manipulation behavior includes changing an instruction set architecture under which instructions are interpreted by the computer.

---

[3] Applicant submits that the change from "the contents of a table entry" to "a content of a table entry" is not a narrowing amendments. Applicant also notes that inherent properties of a claim element do not require independent antecedent basis, MPEP § 2173.05(e), and thus this amendment is not made in response to any statutory ground of rejection.

Applicant respectfully asserts that "control of behavior" and "changing ISA" are "related" because one "includes" the other. For example, a given bit pattern may code for an addition in one ISA, and may code for subtraction, a control transfer, or be an illegal instruction in the other. Thus, the step of changing one may include changing the other. Applicant believes that no amendment is necessary.

K. "alter—in manner incompatible with the architectural definition of the instruction"

The Examiner requested information as to the meaning of "alter — in manner incompatible with the architectural definition of the instruction" as used in claim 10.

As is well known in the art, an architecture defines the behavior of each instruction in its instruction set. For example, most architectures define an "ADD" instruction to cause the addition of two numbers, a "JUMP" instruction to transfer control to another instruction in accordance with defined rules, and similar definitions for all other instructions in the instruction set. To "alter [an instruction's behavior] in a manner incompatible with the architectural definition of the instruction" is to cause an instruction to do something other than its architecturally-defined behavior. For example, an altered "ADD" instruction might perform a subtract, cause a transfer of control, or be an illegal instruction. An altered "JUMP" instruction might cause a control transfer to a destination other than the architecturally-defined destination, and/or cause a change in ISA under which further instructions are executed. Some examples are described in the specification at sections II (pages 44-46), IV (pages 64-73), VI.A (pages 100-101), VI.D (pages 106-111) and VI.E (pages 111-112). The specification may include other examples.

Applicant believes that the plain language of the clause would be will understood by those in the art.

L. "which point of time the table is accessed"

The Examiner requested information as to the "point of time the table is accessed and the determination is initiated."

Claim 11 is dependent on claim 10, and therefore incorporates claim 10 by reference. Claim 10 recites, in pertinent part, as follows:

> table lookup circuitry designed to index into a table by a memory address of a memory reference arising during execution of an instruction, and to retrieve a

table entry corresponding to the address, the table entry being distinct from the memory referenced by the memory reference;

As amended, claim 11 recites in pertinent part:

> pipeline control circuitry is further designed to initiate a determination of whether to transfer control from an execution of the instruction, the instruction being an instruction of the first binary representation of the program, to the second binary representation, and effective to initiate the determination with neither a query nor a transfer of control to the second binary representation being coded into the first binary representation.

Claim 10 recites that the table entry is accessed as a consequence of "a memory reference arising during execution of an instruction." As amended, claim 11 recites that the "determination" is to "transfer control from an execution of the instruction." Thus, the "the table is accessed" and "the determination is initiated," at the very earliest, after the address of the instruction has been generated preparatory to a fetch of the instruction, and, at the very latest, before results of following instructions are irreversibly committed to architecturally-visible machine state. Applicant believes that the claim as previously examined or as now amended would be so understood by one of skill in the art.

For example, see the specification at sections II (pages 44-46), IV (pages 64-73), VI.A (pages 100-101 of the specification), VI.B (pages 101-103), VI.D (pages 106-111) and VI.E (pages 111-112). The specification may contain other examples as well.

M. "relationship recited between interrupt and change of behavior"

The Examiner requested information as to the meaning of the "relationship between interrupt and change of behavior."

Claims 13 and 28 have been amended to make the relationship clearer.[4]

The applicability of the remainder of this paragraph to the claims is not understood. For example, claim 14 recites neither an interrupt nor a "change of behavior." Claim 20 does not recite a "change in behavior."

---

[4] Applicant submits that breaking one clause into two is not a narrowing amendment. Applicant also submits that the claim was sufficiently "clear and definite" before amendment, so that the amendment is not made in response to a statutory ground of rejection.

N. "functional relationship between the components"

The Examiner requested information as to the "functional relationship between the components" of claim 14.

By this paper, Applicant amends claim 14 to address the issue. Applicant believes that the elements of claim 14 are now properly interrelated.

O. "Function of the interrupt handler"

The Examiner requested information as to the meaning of the "Function of the interrupt handler" as recited in claim 21. Claims 20 and 21 recite, in pertinent part:

> 20. The microprocessor chip of claim 19, further comprising:
> interrupt handler software designed to service the interrupt and to return control to an instruction flow of the process other than the instruction flow triggering the interrupt, the returned-to instruction flow for carrying on non-error handling normal processing of the process.
>
> 21. The microprocessor chip of claim 20, wherein the interrupt handler software is programmed to change an instruction set architecture under which instructions are interpreted by the computer.

As the Examiner observes, the interrupt handler of claim 21 does some things that are not "common." Claim 20 recites that the handler is "designed to service the interrupt and to return control to an instruction flow of the process other than the instruction flow triggering the interrupt." Claim 21 recites that "the interrupt handler software is programmed to change an instruction set architecture."

If an interrupt handler is programmed to perform the tasks recited in claims 20 and 21, Applicant intends that the handler be within the respective claims. If the handler is incapable of performing these functions, Applicant intends that it be outside the claims. Applicant believes that these claims, as written, are "particular and distinct," and conform to § 112 ¶ 2.

P. "logically equivalent"

The Examiner requested information as to the meaning of "equivalent" in claim 22, and questioned the antecedent basis for the "two instruction text" in the wherein clause of claim 22.

By this paper, claim 22 has been amended to provide better antecedent basis. This is not a narrowing amendment.

Moreover, at page 100, lines 10-17, the specification discusses one possibility for "logically equivalent" instruction text: RISC code, for example, produced by a binary translator,

D

that performs similarly enough to the original X86 program that the two programs produce the same result.

Accordingly, Applicant believes that claim 22 fully complies with § 112.

Q. Dependency of claim 26

The Office Action notes that the dependency of claim 26 was not clear. By this paper, claim 26 has been amended to improve the clarity of the dependency.

R. "single means claim"

The Examiner questioned claims 24 and 30 on the following basis:

> Claims 24 and 30 are rejected under 35 U.S.C. 112, 1st paragraph, as those claims are impermissible single means claims containing undue breath. (See MPEP 2164.08(a) and 2181).

Applicant believes that this rejection is improper and should be withdrawn. MPEP § 2164.08(a) reads, in pertinent part, as follows:

> **2164.08(a) Single Means Claim**
>
> A single means claim, i.e., where a means recitation does not appear in combination with another recited element of means, is subject to an undue breadth rejection under 35 U.S.C. 112, first paragraph.

MPEP § 2164.08(a) and § 2181 make clear that a "single means" rejection can only arise in the context of a means-plus-function claim. Claims 24 and 30 are method claims, not means-plus-function claims. Accordingly, Applicants believe that MPEP § 2164.08(a) and § 2181 have no applicability to claims 24 and 30.

IV. Requests for identification of components in the drawings and specification

A. "table lookup circuitry"

The Examiner requested information as follows:

> Applicants are requested to identify the following components in the drawings and the description thereof in the specification:
>
> 1. the table lookup circuitry,

One specific example of "table lookup circuitry" is shown in Figs. 6b and 6c, and discussed in section VI.D (pages 106-111). Another example is the circuitry that consults ISA flag bit 180, 182 as discussed in section II (pages 44-46). Another example is the circuitry that consults the CC flag bit 200 in section IV (pages 64-73). Other examples are discussed throughout section

VI of the specification, from pages 100-116, and more generally throughout the specification and figures.

B. "the table with entry having the likelihood description"

The Examiner requested information as follows:

> Applicants are requested to identify the following components in the drawings and the description thereof in the specification: ...
>
> 2. the table with entry having the likelihood description

The PIPM (reference 602) and its bits 624 constitute one such table. PIPM 602 is shown in Fig. 6a, and discussed throughout the specification, particularly in section I.A (pages 29-31), section I.C (pages 35-36), section VI.B (pages 101-103), section VI.C (pages 103-106), section VI.D (pages 106-111), and section VI.G (pages 113-115). Other examples of such tables are discussed in section VI.A (pages 100-101).

C. "interrupt circuitry and the handler"

In the Office Action, the Examiner requested information as follows:

> Applicants are requested to identify the following components in the drawings and the description thereof in the specification: ...
>
> 3. the interrupt circuitry and the handler,

One particular example of the interrupt circuitry and handler are disclosed in Figs. 6b and 6c, and discussed in section VI.D (pages 106-111) of the specification. Other examples are discussed throughout section VI (pages 100-116). Other examples include the hardware and software that handle the ISA flag bit 180 (section II, pages 44-46), and the CC flag bit 200 (section IV, pages 64-73).

D. "the architectural definition of the instruction not calling for an interrupt"

In the Office Action, the Examiner requested information as follows:

> Applicants are requested to identify the following components in the drawings and the description thereof in the specification: ...
>
> 4. the meaning of "the architectural definition of the instruction not calling for an interrupt", and

This is discussed in section III.E of these Remarks.

E. "pipeline circuitry to effect control of instruction behavior"

In the Office Action, the Examiner requested information as follows:

> Applicants are requested to identify the following components in the drawings and the description thereof in the specification: ...
>
> 5. the description of the pipeline circuitry to effect control of an architecturally-visible data manipulation behavior and control transfer behavior.

The Examiner is referred to sections III.F and III.K of these Remarks. One specific embodiment is described in Figs. 6b and 6c, and discussed in section VI.D (pages 106-111) of the specification. Other examples are discussed in section VI.A (pages 100-101), and throughout section VI (pages 100-116).

V. § 103(a) rejections over Morley '982 and Woods '032

The Examiner comments on independent claims 1, 2, 10, 19 and 30 as follows:

> Claims 1-13 and 19-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Morley (5,751,982) in view of Woods (5,678,032).
>
> See at least background of the invention, summary of the invention and the description of Figures 3 and 4 in Morley.
>
> Morley teaches a microprocessor chip, comprising:
>
> 1. An instruction circuitry (CPU line 14, column 1),
>
> 2. A table lookup circuitry (dispatcher),
>
> 3. A table (16) having entries for indicating whether native code 18 or native code 26 should be accessed (therefore result in different behavior),
>
> 4. An interrupt circuitry (the emulator) cooperatively designed with the instruction circuitry to trigger an interrupt (when the emulator encounters an instruction or frequently executed instruction to be emulated).
>
> The only difference is that Morley did not specify that the instruction circuitry is a pipeline one. Instruction pipeline circuitry is well known in the art. Woods teaches an emulator having pipeline processor. Since both references are directed toward emulator and pipeline processor is well known in the art, it would have been obvious to a person of ordinary skill in the art to use a pipeline processor in Morley such that instruction execution can be pipelined.

Morley '982 is roughly similar to the Adachi '975 patent raised in the prior Office Action. In Morley '982, when a region of code is translated from one ISA into logically equivalent code of another, the first instruction in the original program is overwritten with a bit pattern that does not correspond to any legal instruction (Morley '982, col. 5, lines 50-56). Any attempt to execute that illegal bit pattern will cause an illegal instruction trap. The instruction

D

dispatch table in the emulator is updated so that an illegal instruction trap for the bit pattern will cause a dispatch to the translated version of the original program (Morley '982, col. 5, lines 59-63).

Thus, the emulator for the old ISA executes each legal instruction in conformance with its definition of the old architecture. When Morley's emulator encounters one of the illegal bit patterns, the value of the bit pattern (not its address) is used to dispatch to the translated version (Morley '982, col. 5, lines 59-63).

The Woods '032 patent will be discussed below.

A. Claim 2

As amended, claim 2 recites as follows:

> 2. A method, comprising the steps of:
>
> as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, consulting a table, the table being indexed by the address of instructions executed, entries of the table containing attributes of instructions whose addresses index to the respective entries;
>
> controlling an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on a content of a table entry associated with the address of the instruction.

Claim 2 recites that the table is "indexed by the address of instructions executed." The Office Action nowhere addresses this limitation. For this reason, any rejection under § 103(a) may be withdrawn.

Further, dispatch table 16 in Morley '982 is indexed by the value of an instruction, that is, the contents of a memory location (see Morley '982 at col. 4, lines 27-35; col. 5, lines 59-62; col. 5, lines 21-31), not by the address of the instruction (as recited in claim 2). This is analogous to the difference between the claims and the Adachi '975 patent, discussed in the previous Response to Office Action.

Applicant's review of Woods '032 reveals no table indexed by the value of an instruction.

Applicant believes that neither Morley '982, Woods '032 nor their combination teach or suggest the limitations of claim 2.

D

B. Claim 10

Claim 10 recites as follows:

> 10. A microprocessor chip, comprising:
>
> instruction pipeline circuitry;
>
> table lookup circuitry designed to index into a table by a memory address of a memory reference arising during execution of an instruction, and to retrieve a table entry corresponding to the address, the table entry being distinct from the memory referenced by the memory reference;
>
> the instruction pipeline circuitry being responsive to the contents of the table to alter a manipulation of data or transfer of control behavior of the instruction in a manner incompatible with the architectural definition of the instruction.

Claim 10 recites a table that is indexed "by a memory address of a memory reference arising during execution of an instruction." The Office Action nowhere addresses this limitation. For this reason, any rejection under § 103(a) may be withdrawn.

Further, dispatch table 16 in Morley '982 is indexed by the value of an instruction (as discussed in section V.A of these Remarks), not by the "address of a memory reference," as recited in claim 10. Applicant's review of Woods '032 reveals no table indexed "by a memory address of a memory reference arising during execution of an instruction."

Applicant believes that neither Morley '982, Woods '032 nor their combination teach or suggest the limitations of claim 10.

C. Claim 19

As amended, claim 19 recites as follows:

> 19. A microprocessor chip, comprising:
>
> instruction pipeline circuitry; and
>
> interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of an instruction of a process in accordance with interrupt criteria, the interrupt criteria being based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

Even before amendment, claim 19 recited circuitry that "[triggers] an interrupt on execution of an instruction [whose] architectural definition [does] not call for an interrupt" based on certain conditions. The § 103(a) portion of the Office Action nowhere addresses this

limitation. MPEP § 2143.01 states that "All words of a claim must be considered in judging the patentability of that claim against the prior art. ... A claim limitation which is considered indefinite cannot be disregarded." For this reason, any rejection under § 103(a) may be withdrawn.

Further, Morley '982 fails to discuss such interrupt circuitry. Though Morley '982 mentions "interruptions" at several points, no interruptions in Morley '982 arise in the context of interrupting "an instruction ... wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt." let alone based on the conditions recited in the claim.

Applicant's review of Woods '032 reveals no interrupting of "an instruction ... wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt."

Applicant believes that neither Morley '982, Woods '032 nor their combination teach or suggest the limitations of claim 19.

VI. § 103(a) rejection over Morley '982, Woods '032 and Bianchi '029

The Office Action states as follows:

> Claims 14-18 are rejected under U.S.C. 103(a) as being unpatentable over Morley (5,751,982), Woods (5,678,032) further in view of Bianchi (6,006,029).
>
> Morley and Woods disclose claim combination set forth above. Morley does not appear to have an address translator. However, Bianchi discloses an emulator having an address translator (item 5, column 18). It would have been obvious to a person of ordinary skill in the art to incorporated an address translator in Morley if address are required to be mapped or translated.

As now amended, claim 14 recites as follows:

> 14. A microprocessor chip, comprising:
>
> instruction pipeline circuitry;
>
> address translation circuitry; and
>
> a lookup structure having entries associated with corresponding address ranges generated by the instruction pipeline circuitry and translated by the address translation circuitry, the entries describing a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range.

D

Claim 14 recites a table whose entries "[describe] a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address ranges." The § 103(a) portion of the Office Action nowhere addresses this limitation. For this reason, any rejection under § 103(a) may be withdrawn.

Further, none of the prior art tables behave in this manner. Table 16 in Morley '982 contains the address of a translated instruction sequence (Morley '982, col. 5, lines 59-62), not the "likelihood" of whether "an alternate coding of instructions" exists at all, as recited in claim 14. Applicant's review of Woods '032 reveals no table whose entries "[describe] a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address ranges." Bianchi '029 discloses only a fairly conventional address translation scheme, in which the address translation table holds the addresses of memory pools (e.g., col. 28, lines 48-57), not the "likelihood of the existence of an alternate coding of instructions" as recited in claim 14.

Applicant believes that neither Morley '982, Woods '032, Bianchi '029 nor their combination teach or suggest the limitations of claim 14

Claim 24 is patentable for similar reasons.

VII. Dependent claims

The dependent claims, 3-9, 11-13, 15-18, 20-23, 25-29, and 31-35 are purportedly rejected over the art. However, the § 103(a) written rejections are entirely silent with respect to limitations recited in these claims. Such piecemeal examination is discouraged by 37 C.F.R. § 1.105 and MPEP § 707.07(g). MPEP § 2143.03 specifically requires that limitations of dependent claims be considered. Applicant requests that any future Office Action indicate the allowability of any claim that recites a limitation against which no prior art is cited.

VIII. Conclusion

In view of the amendments and remarks, Applicant respectfully submits that the claims are in condition for allowance. Applicant requests that the application be passed to issue in due course. The Examiner is urged to telephone Applicant's undersigned counsel at the number noted below if it will advance the prosecution of this application, or with any suggestion to resolve any condition that would impede allowance. In the event that further extension of time is required, Applicant petitions for that extension of time required to make this response timely.

Kindly charge any additional fee, or credit any surplus, to Deposit Account 50-0324, Order No. 30585/16.

Respectfully submitted,

SHEARMAN & STERLING

Dated: August 10, 2001

By: _____
David E. Boundy
Registration No. 36,461

Mailing Address:
SHEARMAN & STERLING
599 Lexington Avenue
New York, New York 10022
(212) 848-4000
(212) 848-7179 Telecopier

D

1. A microprocessor chip, comprising:

instruction pipeline circuitry; and

table lookup circuitry designed to retrieve an entry from a table, each entry of the table being associated with a corresponding address range translated by address translation circuitry of the microprocessor chip, each entry describing a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range, the table lookup circuitry operable as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer;

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to synchronously trigger an interrupt in accordance with interrupt criteria on execution of an instruction of a process, wherein the architectural definition of the instruction does not call for an interrupt, the interrupt criteria being based at least in part on the table entry associated with the address of the instruction, the interrupt circuitry being designed to invoke a handler for the interrupt, the handler being responsive to a content of the table entry to affect the instruction pipeline circuitry to effect control of an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on the contents of a table entry associated with the address range in which the instruction lies.

2. A method, comprising the steps of:

as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, consulting a table, the table being indexed by the address of instructions executed, entries of the table containing attributes of instructions whose addresses index to the respective entries; and

controlling an architecturally-visible data manipulation behavior or control transfer behavior of the instruction based on a content of a table entry associated with the address of the instruction.

3. The method of claim 2, wherein the control of control transfer behavior includes transfer of execution control to a second instruction for execution.

4. The method of claim 3, wherein the second instruction is coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

7. The method of claim 2, wherein:

entries of the table describe a likelihood of the existence of an alternate coding of instructions located in respective corresponding address ranges.

8. The method of claim 2, wherein:

entries of the table correspond to pages managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is integrated with virtual memory address translation circuitry of the computer.

9. The method of claim 2, further comprising the steps of:

synchronously triggering an interrupt on execution of an instruction of a process in accordance with interrupt criteria, the interrupt criteria being based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

11. The microprocessor chip of claim 10, further comprising:

a binary translator programmed to translate at least a selected portion of a computer program from a first binary representation to a second binary representation; and

within the pipeline control circuitry is further designed to initiate a determination of whether to transfer control from an execution of the instruction, the instruction being an instruction of the first binary representation of the program, to the second binary representation, and effective to initiate the determination with neither a query nor a transfer of control to the second binary representation being coded into the first binary representation.

12. The microprocessor chip of claim 10, further comprising:

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of the instruction in accordance with interrupt criteria, the interrupt criteria being based at least in part on a memory state of the computer and the

address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

13. The microprocessor chip of claim 12, further comprising:

interrupt handler software designed to service the interrupt and to effect the altering of a transfer of control behavior, the altering being in the form of returning control to an instruction flow of the process other than the instruction flow triggering the interrupt, the returned-to instruction flow for carrying on non-error handling normal processing of the process.

~~14~~ 18. A microprocessor chip, comprising:

instruction pipeline circuitry;

address translation circuitry; and

a lookup structure having entries associated with corresponding address ranges generated by the instruction pipeline circuitry and translated by the address translation circuitry, the entries describing a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range.

18. The microprocessor chip of claim 14, further comprising:

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of an instruction of a process, synchronously based on interrupt criteria, the interrupt criteria based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

19. A microprocessor chip, comprising:

instruction pipeline circuitry; and

interrupt circuitry cooperatively designed with the instruction pipeline circuitry to trigger an interrupt on execution of an instruction of a process in accordance with interrupt criteria, the interrupt criteria being based at least in part on a memory state of the computer

and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

26 22. The microprocessor chip of claim 24 20, wherein the returned-to instruction flow is logically equivalent to the instruction flow beginning at the interrupted instruction.

30 24. A method, comprising the steps of:

as an integral part of processing an instruction in instruction pipeline circuitry of a computer, consulting a lookup structure of entries, entries of the lookup structure corresponding to address ranges translated by address translation circuitry, and the entries describe a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address ranges.

26. The method of claim 24, further comprising the step of:

altering a behavior of the instruction in a manner incompatible with the architectural definition in an emulated architecture of the instruction, based on a content of the entry corresponding to the address range containing the instruction.

27. The method of claim 24,

wherein each entry corresponds to a page managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is integrated with virtual memory address translation circuitry of the computer.

28. The method of claim 24, further comprising the step of:

based on a content of the entry, transferring control to the alternative coding, the alternative coding being an instruction flow of the process other than the instruction flow triggering the consulting, the returned-to instruction flow being programmed to carry on non-error handling normal processing of the process.

29. The method of claim 24, further comprising the step of:

based on a content of the entry, synchronously triggering an interrupt in accordance with interrupt criteria, the interrupt criteria being based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition in an emulated architecture of the instruction does not call for an interrupt.

30. A method, comprising the steps of:

on execution of an instruction of a process, synchronously triggering an interrupt based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction does not call for an interrupt.

34. The microprocessor chip of claim 30, further comprising:

table lookup circuitry designed to index into a table by a memory address of a memory reference arising during execution of an instruction, and to retrieve a table entry corresponding to the address;

the instruction pipeline circuitry being responsive to a content of the table to affect a manipulation of data or transfer of control defined for the instruction.

36. The microprocessor chip of claim 10, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

37. The microprocessor chip of claim 19, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

38. The method chip of claim 30, wherein the architectural definition of the instruction with which the alteration is incompatible is a definition in an emulated architecture.

39. A method, comprising the steps of:

as part of the basic instruction cycle of executing an instruction of a non-supervisor mode program executing on a computer, retrieving an entry from a table, the entry of the table being indexed by the address of a memory reference arising during execution of the instruction, the table entry being distinct from the memory referenced by the memory reference;

based on a content of the table entry, altering a manipulation of data or transfer of control behavior of the instruction in a manner incompatible with the architectural definition in an emulated architecture of the instruction.

~~40~~ 44. The method of claim ~~39~~ 43, wherein the entries of the table correspond to ranges of memory addresses.

41. The method of claim 39, wherein the entry of the table is indexed by the address of instructions executed.

42. The method of claim 39, wherein:

entries of the table correspond to address ranges, and the entries describe a likelihood of the existence of an alternate coding of instructions located in the respective corresponding address range.

43. The method of claim 39:

wherein the architectural definition of the instruction in the emulated architecture does not call for an interrupt;

and further comprising the step of synchronously triggering an interrupt based at least in part on a memory state of the computer and the address of the instruction.

44. The method of claim 39, wherein the control of transfer of control behavior includes transfer of execution control to a second instruction for execution.

45. The method of claim 44, wherein the second instruction is coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

46. The method of claim 39, wherein the control of architecturally-visible data manipulation behavior includes changing an instruction set architecture under which instructions are interpreted by the computer.

47. The method of claim 39, wherein the behavior control includes selecting between two different instruction set architectures, and the computer includes instruction pipeline circuitry designed to effect interpretation of computer instructions under the two instruction set architectures alternately.

48. The method of claim 39:

wherein each entry of the table corresponds to a page managed by a virtual memory manager, and wherein circuitry for locating an entry of the table is integrated with virtual memory address translation circuitry of the computer.

49. The method of claim 39, further comprising the steps of:

triggering an interrupt on execution of an instruction of a process, synchronously based at least in part on a memory state of the computer and the address of the instruction, wherein the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

50. An apparatus, comprising:

instruction pipeline circuitry; and

table lookup circuitry designed to retrieve a table entry from a table indexed by an address of an instruction fetched for execution;

the instruction pipeline circuitry being responsive to a content of the table entry to control an architecturally-visible data manipulation behavior or control transfer behavior of the fetched instruction based on a content of the table entry associated with the address of the instruction.

51. The method of claim 50, wherein the control of control transfer behavior includes transfer of execution control to a second instruction for execution, the second instruction being coded in an instruction set architecture (ISA) different than the ISA of the executed instruction.

52. The method of claim 50,

wherein entries of the table correspond to pages managed by a virtual memory manager, circuitry for locating a table entry being integrated with virtual memory address translation circuitry of the computer.

54. The method of claim 50, further comprising the steps of:

triggering an interrupt on execution of an instruction of a process, synchronously based at least in part on a memory state of the computer and the address of the instruction, when the architectural definition of the instruction in an emulated architecture does not call for an interrupt.

55. The method of claim 50, wherein:

the table entries are indexed by a virtual address of the instruction.

56. The method of claim 50, wherein:

the table entries are indexed by a physical address of the instruction.

~~57~~. The method of claim ~~24~~, further comprising the step of:

after determining that the existence of an alternate coding is likely, consulting a second table, the entries of the second table definitively indicating entry points for initiating such alternate codings as exist.